SPARQL Query Optimization using Data Structure Approach

Ms Meela D Gareja[1]
[1]ME Student
[1]Department of Computer Engineering
[1]Nobel Group of Institutions- Junagadh

Abstract— Linked open data also known as web of linked data is a globally distributed database. This linked data can be queried with SPARQL protocol and RDF query language also known as SPARQL. Our key goal for this topic is to optimize the SPARQL engine with help of basic data structure algorithms. Our approach proposed the optimizing SPARQL engine by cleaning the Ontology file by loss some loss of knowledgebase and the compare the results of SPARQL query processing time on both Ontologies one is cleaned and other one is without cleaned Ontology. We apply the data structure searching algorithms to minimize search space. And after cleaning approach we can optimize loading time of our ontology. The paper gives a brief understanding for the proposed approach.

Key words: Linked Data, SPARQL, RDF, Pruning

I. INTRODUCTION

Tim Berners-Lee in May 2001 described his vision of a "new form of web content that is meaningful to Computers" and would "unleash a revolution of new possibilities" in an article called The Semantic Web. Main idea of Semantic Web is encoding knowledge in machine readable format, in coexistence with human-readable HTML pages which are almost impossible to understand and interpret by computer. The goal was that the semantic annotation of Web resources would allow computers to automatically reason about the underlying data and enable them to make reliable and logically founded conclusions. At a global scale, the Semantic Web thus can be understood as a huge knowledge base that links information from different resources together, eases the access to knowledge, and ultimately improves search and knowledge discovery in the Internet.

One research area of the linked open data project is querying the linked data. Querying this linked data efficiently is essential with the growth of the linked data cloud. Linked data is always in the form of Resource Description Framework (RDF) format. After the development of different prototypes for querying this cloud structured data SPARQL protocol and RDF query Language was developed and accepted by the World Wide Web consortium as the query language to query Linked data.

This optimization is required due to the growth of the data cloud which will grow even more eventually with more websites converging towards the Semantic Web.

II. MOTIVATION AND SCOPE

RDF, as a free-schema data model, offers a seamless integration of datasets allowing the exchange and processing of knowledge using automated intelligent methods [8]. On the other hand, due to its inherent graph-structure, there is requirement of efficient approach or mechanisms for querying large RDF datasets to speed up the retrieval of information.

Examples of huge datasets are, for instance, the UniProt database containing more than 600 million triples [9] or the W3C SWEO Linking Open Data Community with more than 4 billion triples [10]. With such datasets, executing SPARQL queries becomes a problem. Note that the execution time is heavily influenced not only by the size of the dataset (number of triples) but also, as stated before, by the number of joins required to find them.

A. Approach I- Federation:

Basic Idea of federation can be described as, evaluation of queries against distributed sources by splitting the query in appropriate sub-queries and combining the results from the remote sources [8].

Federation came into being due to the anomaly posed by integration of meta-data into a central repository. This may not be possible each and every time due to the large amount of distributed data available on the cloud. Thus federation over multiple repositories and SPARQL endpoints was proposed.

Single repositories are federated into a federation layer, which splits the query into sub-queries against the single repositories and produces results. These results are then merged together [8]. To the user this approach is completely transparent that is no query rewriting is needed to support this feature.

In using SPARQL endpoints, multiple single repositories are accessed using a federator. This approach deletes the step of loading data and instead a SPARQL endpoint can be made for the same. Disadvantage is, SPARQL endpoint is restricted and essential statistical information is not available which is needed for query optimization [8].

Another problem faced is, the multiple results produced at each stage of execution. Federation will produce even more results at each stage then a normal query execution. Such results need to be evaluated for their relevance and the irrelevant sub graphs should not be explored. This is where the described Approach 2 helps.

III. APPROACH II- CONSTRAINT BASED PRUNING

As described in the previous approach there is a need for pruning results which are not relevant at each stage of execution. According to [2] a SPARQL query is executed by iteratively dereferencing URIs. RDF descriptions are fetched from the web and solutions are built based on Retrieved data.

For example:

```
SELECT ?friend ?entity ?number WHERE
{
http://site/Chirag_P.rdf m1:hasFriend ?friend.
?friend nm1:worksFor ?entity.
?entity univ:numberOfStudents ?number.
}
```


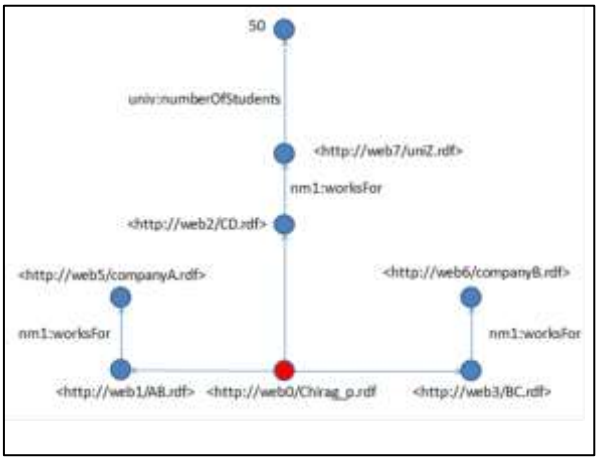

Fig. 1: Query Exploration For Example Query

Here there is no need to executed viz. explore the sub-graphs of friends AB and BC as they work for consultancy companies which will never have the entity named students. This can be determined by meta-data or vocabularies at the SPARQL endpoint. Only results for CD are needed.

This can be realized by use of probability and selectivity of results. The proposal is to express the data as a context graph and develop an algorithm which will score the results at each stage of the execution based on probability and selectivity of the results. The selectivity of the results can be calculated based on threshold values predetermined based on available meta- data, vocabularies and access to SPARQL endpoints. The probability function developed will help produce selectivity scores of each result. These scores can be compared to the threshold and results can be pruned, thus decreasing sub-graph exploration of non-relevant results at each stage. Theoretically this approach proves to be of great improvement over the existing query execution methods. A pictorial representation of the query graph exploration after application of proposed approach is shown in Figure 2.

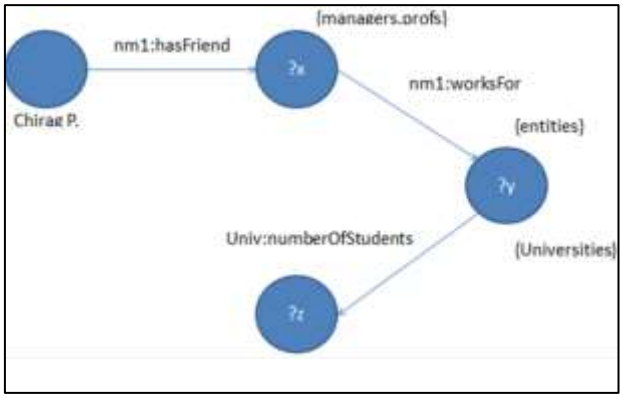

Fig. 2: Query Graph Exploration after Application of Proposed Approach

IV. Proposed Approach

We optimize the various factors like Load time of data sets, query Optimization. For that step by step explained below

1) Step 1: Load the data set i.e. OWL.
2) Step 2: Convert the OWL in Graph format using various parsing API.
3) Step 3: Remove the nodes (Knowledgebase) which does not require for query processing.
4) Step 4: Generating optimized Ontology file after pruning the nodes.
5) Step 5: Optimize the SPARQL query plan.
6) Step 6: SPARQL query is executed on previous data set and optimized data sets.
7) Step 7: Comparing result of normal query execution and optimized query execution.

V. Conclusion and Future Work

From the study we can conclude that we can optimize the SPARQL query using this novel approach. So many approaches is proposed but very few of that is implemented. And from literature survey we can found that our approach is overcome current problems of semantic web. By removing redundant node we can optimize the load time of ontology so SPARQL engine is optimized. And by reducing search space reduce query time for SPARQL query execution, and our approach is generic for all ontology.

In future we will implement this system using JAVA language and Different API for ontology. We will implement loading ontology and converting it Graph presentation.

References

[1] Bizer, C., Heath, T., Berners-Lee, T.: Linked data –the story so far, International Journal On Semantic Web and Information Systems,2009
[2] Florian Bauer, Martin Kaltenbock:linked Open Data-The Essentials
[3] Ladwig, G., Tran, T.: Linked Data Query Processing Strategies. In: Patel- Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 453–469. Springer, Heidelberg (2010)
[4] Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.-U., Umbrich, J.: Data summaries for on-demand queries over linked data. In: WWW (2010)
[5] Hartig, O., Bizer, C., Freytag, J.-C.: Executing SPARQL Queries over the Web of Linked Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 293–309. Springer, Heidelberg (2009)
[6] Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: A Federation Layer for Distributed Query Processing on Linked Open Data. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. eds.) ESWC 2011. LNCS, vol. 6644, pp. 481–486. Springer, Heidelberg (2011)
[7] G¨orlitz, O., Staab, S.: Federated Data Management and Query Optimization for Linked Open Data. In: Vakali, A., Jain, L.C. (eds.) New Directions in Web Data Management 1.zSCI, vol. 331, pp. 109–137. Springer, Heidelberg (2011)
[8] Kudeep Reddy & Sreenivasa Kumar: Optimizing SPARQL Queries Over The Web Of Linked Data, 2010 [1]

